doi:10.3772/j.issn.1002-0470.2015.07.003

片上多核处理器的区域共享的双粒度目录①

曾 露②[*,**,***] 陈新科[*,**,***] 王焕东[****]

([*]计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)

([**]中国科学院计算技术研究所 北京 100190)

([***]中国科学院大学 北京 100049)

([****]龙芯中科技术有限公司 北京 100190)

摘 要 研究了双粒度目录(DGD)下片上多核处理器的访存行为以及DGD对不同共享行为的目录开销,以进一步降低DGD结构的面积开销。针对DGD需要为共享缓存区域创建额外的块目录项的问题,提出了创新的区域共享的双粒度目录(RSDGD)结构。该结构可用一个区域共享目录项同时维护最多3个共享者共享同一个缓存区域的一致性,从而能有效减少所需的块目录项数量,降低总的目录开销。实验结果表明,和原有的DGD相比,该结构平均减少了25%的目录空间需求,而仅产生了不到0.6%的性能损失。该结构有效地降低了芯片的面积开销,提高了目录结构的可伸缩性。

关键词 双粒度目录(DGD),片上多核处理器,缓存一致性,区域共享,目录一致性协议,访存优化

0 引 言

多核/多片互连处理器的设计广泛采用基于目录的缓存一致性协议。随着处理器核心数量的增多和私有缓存容量的增大,需要为工作集提供足够的目录项以保证缓存一致性协议的性能,从而目录项数呈线性增长。同时,共享者增多导致目录项需要提供额外的存储空间用以维护缓存一致性信息。综合上述两个方面,目录结构在多核/多片处理器中的面积开销呈平方量级增长。因此,优化目录空间,提高目录的空间利用率,成为设计高可伸缩性的缓存一致性协议的必要条件。本文在研究目录空间优化的基础上提出了一种区域共享的双粒度目录(region shared dual-grain directory, RSDGD)结构,该结构比原有双粒度目录(DGD)结构的性能有明显提高。

1 相关工作

目录空间的优化策略主要归纳为两类:降低目录项大小和减少目录项数目。

已有的大量工作如粗向量目录[1]、有限指针目录[2]、分段向量目录[3]、Tagless目录[4]和基于特殊哈希算法的目录结构[5,6]等,通过压缩目录项的共享信息来换取空间。这些方案通常受限于共享者的数量,当共享者超过一定数量时,目录的性能会有所降低。

稀疏目录(sparse directory, SPDIR)[7]的提出使目录的项数不再依赖于内存大小,而是片内缓存的大小。最近的一些工作[8-12]基于稀疏目录的结构进一步减少对目录项数的需求:文献[8]提出的WayPoint在内存中维护了片上目录的后备目录,从

① 国家"核高基"科技重大专项课题(2009ZX01028-002-003, 2009ZX01029-001-003, 2010ZX01036-001-002, 2012ZX01029-001-002-002),国家自然科学基金(61221062, 61100163, 61133004, 61173001, 61232009, 61222204)和863计划(2012AA010901, 2012AA011002, 2012AA012202, 2013AA014301)资助项目。

② 男,1987年生,博士生;研究方向:计算机系统结构;联系人,E-mail: zenglu@ict.ac.cn

(收稿日期:2015-01-26)

而减少了片上目录;文献[9]提出借助操作系统检测私有页和只读共享页,去除目录对这些页的缓存一致性支持,该方案减少了大量不必要的目录项开销,然而它需要修改底层操作系统的内存管理以及终端管理模块,带来了使用维护上的大量开销。文献[10]通过仅将私有和只读的共享缓存块置于 L1 缓存,而将可读写的缓存块仅存于 L2 共享缓存中,来消除记录共享块所需的目录项,该方案同样减少了目录的面积开销,但是由于可读写的缓存块被固定在 L2 缓存中,较大地增加了可读写的缓存块的访问延时。

这些方案均注重于检测出私有块和只读共享块,并取消对这些块的缓存一致性信息的追踪,从而达到减少目录项数的目的。

文献[11,12]基于上述思想和对地址连续共享块访存行为的进一步考察,提出了临时私有缓存区域的概念。通过双粒度目录(dual-grain directory, DGD)结构将多个连续的临时私有块组织到一个目录项中,组成临时私有区域,可降低面积开销。

本文在 DGD 的基础上提出了一种更完善的方案,使用专门的目录项处理多个共享者共享一个缓存区域中的多个块的情形,进一步降低了 DGD 所需的目录项数,提高了目录的空间利用率。

2 双粒度缓存一致性

本文使用片上众核处理器结构来评估双粒度缓存一致性协议的设计。如图 1 所示,作为一个基础架构,处理器由16个相同的处理器核心组成,也可

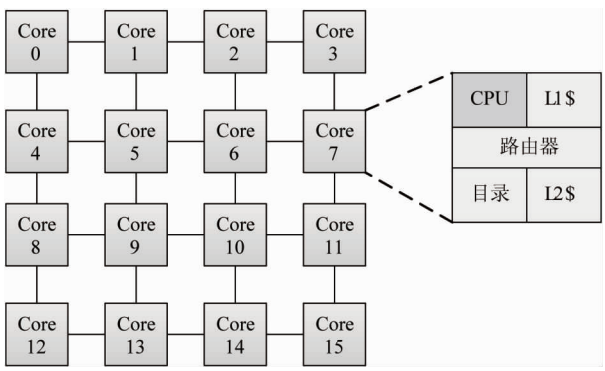

图 **1** 片上众核处理器基本架构

扩展为 64 核或更多核心的结构。处理器核之间通过 Mesh 结构的互连网络连接,组成4 ×4 的 Mesh 网络。每个核包括一个按序发射的处理器核,一个私有的 L1 指令和数据缓存,一个一致性目录缓存和一块共享的 L2 缓存。处理器对 L2 缓存的访问通过专门的哈希函数映射到不同的节点上,从而在网络上分摊访存压力。

通常情况下,共享存储的访存行为可以划分为私有访问、共享只读访问和共享读写访问。基于这种访存行为的划分,现有的目录优化策略包括不维护私有访问块的一致性,或者对于共享读写块只维护其拥有者信息,从而降低目录开销。而对于共享只读访问块,通常采用简单的优化策略统一处理,如有限指针、粗向量等。

共享只读访问空间虽然被多个共享者共享访问,但是观察其整个生存期(即第一个共享者访问该内存到最后一个共享者从私有缓存中替换出来为止)大部分时候仅仅被一个共享者所共享,即在前一个共享者替换回共享缓存后,后一个共享者才访问该块。

因此需要对共享访问空间进行更细致的划分。我们将一段时间内仅被一个处理器核访问并在其被替换出私有缓存前没有其他共享者访问的共享内存区域定义为临时私有 (temporal private, TP)区域。相反,在一段时间内被多个处理器核访问并缓存的共享内存区域被称为临时共享 (temporal shared, TS)区域。这两种区域在物理上可以是同一块缓存区域,但是考虑到时间上的因素,TP 区域会由于随后有新的访问者从而转换为 TS 区域,而 TS 区域也可能由于共享者替换转换为 TP 区域。

已有的研究表明,共享只读访问空间在绝大多数时间表现为 TP 属性,仅在少数情况下同时被多个共享者同时缓存。双粒度目录(DGD)利用这个特性,将地址连续的 TP 块以区域的方式组织,只使用一个区域目录项来维护区域的一致性信息,从而降低目录开销。

DGD 基于稀疏目录(SpDir)结构,在其基础上增加了区域缓存共享目录项的支持。如图 2 所示,通过在目录项中增加一个 R(region)位,用来区分该

目录项是区域目录项还是普通块目录项;而块目录项用以存储共享信息的空间(如位向量)在区域目录项中转换为存在位(present bit),即记录缓存区域中的哪一个缓存块被区域拥有者缓存。

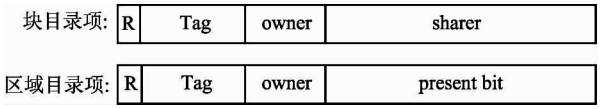

图 2　**DGD** 目录项的字段定义

当处理器第一次访问一个内存区域时,DGD 首先为其创建一个区域目录项(此时该区域目录项应为 TP 区域),并且设置该区域目录项的所有者为请求者。当后续该请求者的内存访问命中了该内存区域,则为其在区域目录项中设置相应的存在位,表示该区域中的这个缓存块被区域所有者所共享,此时位向量的某一位表示区域中的某一个块被区域拥有者访问。

当有其他共享者对同一区域进行共享访问时,DGD 为其额外创建一个单独的块目录项,记录新的请求者与原拥有者为该缓存块的共享者并将原区域目录项中相应的存在位清除。该目录项的位向量此时表示具体的共享者,即原拥有者和新的共享者。

以 1KB 区域大小和 64B 块大小为例,一个区域目录项需要 16 比特位空间(1KB/64B)来保存缓存块的存在位信息。

此外 DGD 还使用了多路命中的方式,以同时查找块目录项和区域目录项。对于某个内存区域,块目录项与区域目录项被哈希算法映射到不同的路,通过一次目录访问,即可同时命中块目录项与区域目录项。

DGD 所带来的创新点主要有以下两点:首先,DGD 提出了缓存块共享行为的时空特性,即存在大量的共享缓存块,其首先被一个处理器访问,在它被替换出私有缓存之前不会被第二个处理器访问。其次,DGD 基于共享缓存块的流式特性,上述行为扩展为内存区域(如 1K 字节),可以在得到相近性能的情况下大量减少所需的目录项数。

然而,尽管 DGD 能够高效地处理私有访问和临时私有 TP 的共享访问,但是它需要为临时共享 TS 的访问请求创建单独的块目录项。随着处理器核心数目的增加以及互连网络复杂度的提升,多个处理器同时访问某块内存区域的几率增加,从而导致 DGD 需要为共享的数据块额外创建块目录项,增加了 DGD 的系统开销,最坏的情况甚至会使 DGD 的性能比 SpDir 还差。开销的增大将对设计高可伸缩性的双粒度目录结构提出新的挑战。针对这种挑战,本文提出了创新性的区域共享的双粒度目录(RSDGD)结构,该结构可有效降低处理 TS 区域的访存开销。

本研究通过观察区域访存行为发现,虽然大多数访存的内存区域通常仅被 1 个共享者共享,但 DGD 为多个共享者访问的内存区域额外创建大量的块目录项,使用区域目录项和块目录项的组合带来了额外的目录项开销。从所需创建的目录项角度来看,二三个共享者共享一个缓存区域所带来的开销相当可观。本文提出的 RSDGD 结构使用一个区域目录项表示多个共享者对该区域的共享访问,而不需要创建额外的块目录项。RSDGD 有效地降低了 DGD 所需要创建的块目录项数,从而总体降低了所需的目录项数,而且加速了区域目录项的一致性维护过程,进一步提升了 DGD 的性能。

3　区域共享的双粒度目录

3.1　访存行为分析

由于引入了内存区域的概念,在考察访存行为时,除了分析共享者的共享行为,还要考察区域内缓存块的访问行为,且这两个因子互为正交。因此本文在分析访存行为时,在考察一个缓存区域在目录中的生存期时,不仅需要考察该区域被多少共享者访问,还需要考察该区域中有多少缓存块被访问过。在这两个条件下,可以得出目录需要为其创建多少个目录项(包括区域目录项和块目录项)。

以 16 个处理器核、1KB 区域空间、64B 块大小的参数配置为例进行分析。如果某区域仅有一个共享者且有 10 个缓存块被访问过,那么对于通常的 SpDir 目录,需要 10 个目录项来维护一致性信息。而 DGD 就仅需一个目录项即可维护一致性信息。

图 3 有助于行为分析,该图中每个测试程序有两组数据,第一组数据表示在缓存区域的生命期内,分别被 1,2,3 以及 4 ~ 16 个共享者访问的缓存区域的数量的百分比,第二组数据表示在缓存区域的生命期内,分别被 1,2,3 以及 4 ~ 16 个共享者访问的缓存区域中需要创建的目录项数的百分比。

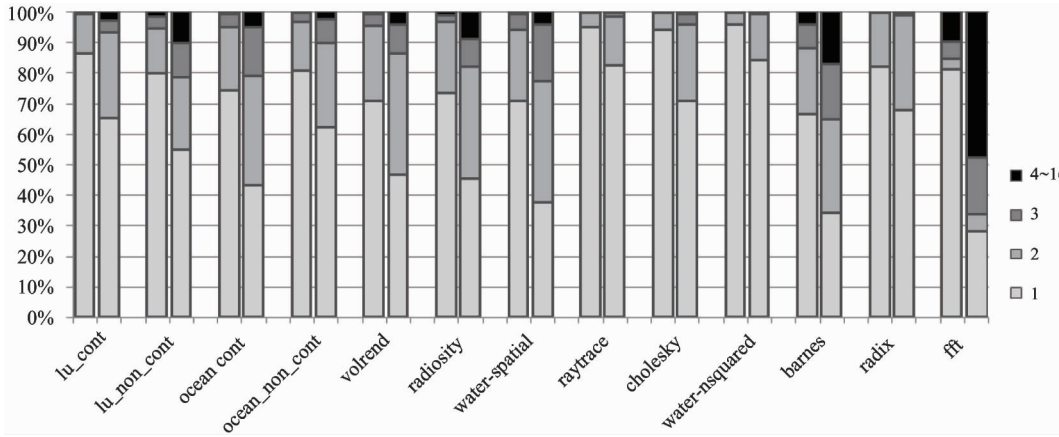

图 **3** **DGD** 的访存行为分析

可以观察到,第一组数据中,共享者数量为 1 的缓存区域占有较大比例,不同测试程序使得该情形所占的比例分布在 66% ~94% 范围。共享者数量为 2 的缓存区域比例约为 4% ~25%;共享者数量为 3 的缓存区域占比约为 0.2% ~5% 范围。超过 3 个共享者的缓存区域通常在 1% 以下。DGD 针对大多数情形(即一个共享者的情形)进行了优化,DGD 仅需一个区域目录项即可维护该缓存区域的一致性,而不需要额外创建块目录项。

对于有两个共享者访问了该区域中的 1 ~16 个缓存块,DGD 除了创建区域目录项外,还需要为第二个共享者访问的每一个缓存块创建一个块目录项。根据第二个共享者所访问该区域中缓存块的数量 $k(1 <= k <= 16)$, DGD 还需要为其创建 k 个目录项。三个共享者共享访问一个缓存区域的情形与之类似,不过相比两个共享者的情形平均所需创建的目录项数更多。

虽然一个共享者的缓存区域情形所占比例较多,不过 DGD 仅需一个目录项即可维护一致性,所需的代价较小。而对于多个共享者的缓存区域来讲,不同访存情形下可能需要 2 ~17 个目录项的开销。根据测试应用的实际运行结果,第二组数据将不同共享者数目下所需创建的目录项数作为考察目标,发现之前占比较少的多个共享者的情形在第二组数据中相应的被放大,如 2 ~3 个共享者的情形占比达 30% ~60%。DGD 方案并没有很好地处理多个共享者访问一个缓存区域的情形,而这种情形虽然在数量上所占比例并不多,但是如果考虑到需要为其创建的额外目录项开销,可以观察到 2 ~3 个共享者共享一个缓存区域的情形所占的目录开销较大,对其优化则显得非常必要。

DGD 能有效处理仅有一个共享者共享缓存区域情形下的缓存一致性,然而对于 2 ~3 个共享者共享一个缓存区域的情形,该区域中可能有 1 ~16 个缓存块被访问,DGD 需要为这些情形创建 2 ~17 个目录项,其中包括首先创建的区域目录项和之后由于其他共享者访问而创建的 1 ~16 个块目录项。可以看到,在最坏的情形下,DGD 需要创建 17 个目录项,而 SpDir 甚至也只需要 16 个目录项。

在同样访存行为下创建更少的目录项,将降低目录结构所需的目录项数,从而减少由于目录替换导致对私有缓存的反向无效,最终降低了片内面积和功耗。本文在 DGD 的基础上提出了区域共享的双粒度目录 RSDGD 优化处理 2 ~3 个共享者访问的区域,降低其所需创建的目录项,从而优化了目录的性能和面积。

如图 4 所示,RSDGD 和 DGD 的主要区别在于对非区域拥有者访问区域目录项时的行为。不同于

DGD 需要创建新的块目录项,RSDGD 将区域目录项转换为区域共享目录项,并记录多个共享的区域访问。在区域共享目录项溢出的情况下,才创建新的块目录项。

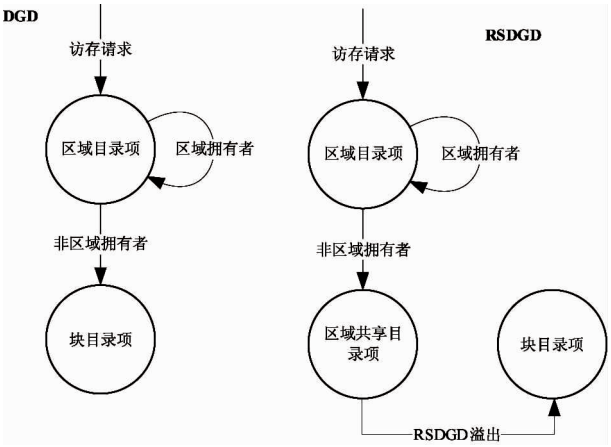

图 4　**DGD** 与 **RSDGD** 的访存行为处理过程

3.2　目录结构

与 Cache 结构类似,DGD 与 SpDir 的目录项通常为多路组相联结构,不过每一个目录项仅维护 Tag 和一致性信息。DGD 在 SpDir 位向量块目录项的基础上增加了区域目录项格式,通过目录项中的 Region 位区分。区域目录项使用 Owner 字段表示缓存区域拥有者,使用存在位向量(present bit vector)表示一个缓存区域中被拥有者所持有的缓存块。本文提出的 RSDGD 在保留前两种目录项格式下增加了区域共享目录项。该目录项的格式如图 5 所示

Tag	sharer1	cnt1	sharer2	cnt2	sharer3	cnt3

图 5　区域共享目录项的字段定义

该图表示 RSDGD 提出的区域共享目录项,用于处理多共享者共享同一区域的情形。Sharer1 – 3 表示最多 3 个区域共享者,cnt1 – 3 表示该拥有者缓存了区域中的缓存块的数量,当 cnt 为 0 表示该共享者字段为空,可以设置新的共享者。

理想情形下,对于区域共享的目录项,需要为每一个共享者记录完整的存在位信息。由于目录组相联结构的特点,专门为区域共享目录项提供更大的空间在设计上并不易于实现,而且会带来更大的存储空间开销。实验数据表明,通过计数器记录拥有者缓存数据块的数量在正确维护一致性信息的前提下并没有带来较大的性能损失。

3.3　缓存一致性行为

DGD 默认会为第一个区域访问者创建一个区域目录项,并设置其为区域拥有者。该访问者后续对该区域其他缓存块的访存都会通过该区域目录项记录一致性信息。当非区域拥有者访问该缓存区域中的某一块地址时,则为其创建一个额外的块目录项,块目录项中记录访问者和区域共享者为该缓存块的共享者,区域目录项中该缓存块所代表的存在位被清零。

之前的访存行为分析证明了 DGD 直接为非区域拥有者创建块目录项带来了大量的性能与面积开销。RSDGD 增加了区域共享目录项的支持,当有非区域共享者访问该缓存区域并且命中了之前创建的区域目录项时,该区域目录项转换为区域共享目录项,同时记录这两个共享者并设置每个共享者所缓存的区域内缓存块的个数。同样,当有第三个访问者命中该区域共享目录项时,设置 sharer3 为该访问者的 id 并设置 cnt3 为 1。之后所有这三个共享者对该缓存区域的访问都无需额外的目录项,只需在区域共享目录项的相应计数器上加一。

当区域共享目录项的三个共享者均已被设置,后续的访问者将类似 DGD 创建一个单独的块目录项来维护该缓存块的一致性。由于区域共享目录项仅记录了共享者缓存块的个数,并不知道某个共享者具体缓存了区域中的哪个缓存块,因此对于这种情况,新创建的块目录项将直接设置这三个共享者作为该缓存块的共享者,无论其是否真的共享该缓存块。而对于当前的区域共享目录项的计数则保持不变。

图 6 给出了 RSDGD 处理访存请求示例。如图 6(a)的示例,当有一个非区域拥有者 5 命中了区域目录项时,RSDGD 会将区域目录项转换为区域共享目录项。由于区域目录项中的存在位有 3 个被置为 1,区域共享目录则将区域拥有者 2 的计数器设置为 3。同时,新增加的区域共享者 5 的计数器被设为

1,表示共享者 5 缓存了该区域中的一个缓存块。

如图 6(b)的示例,当区域共享目录项中的三个共享者都被设置的情况下,有新的请求者命中该项会导致区域共享目录项溢出。由于没有记录共享者 2,5,8 是否包含请求者所请求的缓存块,新创建的块目录项中将这些共享者猜测为共享状态,以避免出现共享信息丢失。同时区域共享目录项中的计数器保持不变,因为区域共享目录项中计数器记录的共享块可能不包含当前请求的共享块,如果在计数器中减去,可能会导致一致性信息记录错误。

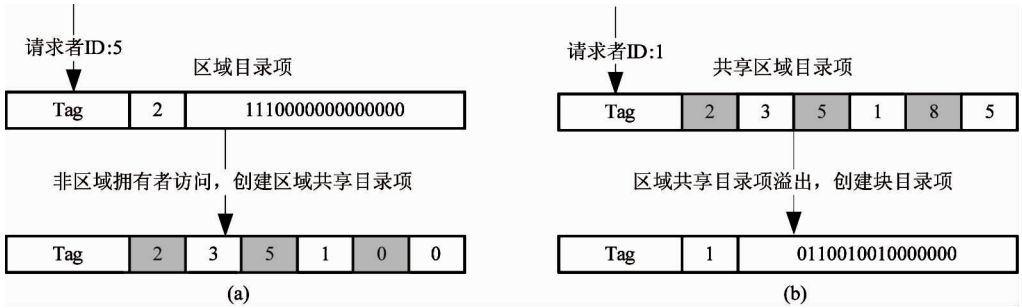

图 **6** **RSDGD** 处理访存请求示例

对于从私有缓存替换出来的缓存块,如果命中了区域共享目录项,那么相应的共享者的计数器应减一。当计数器减为 0 后,该共享者已经没有共享该缓存区域的缓存块,区域共享目录项中该共享者的信息将被清空,可用来记录其他的共享者的共享信息。

上述的一致性行为要求私有缓存能够处理无效请求不命中的情形,即使私有缓存中没有该缓存块,当收到目录对该缓存块的无效请求时,告知目录该缓存块不存在,而不是直接丢弃该请求。因为在创建块目录项的过程并不清楚区域共享者是否共享了该缓存块,因此为了保证一致性协议的正确性,直接假定了该缓存块被区域共享者共享。

然而由于区域共享目录项没有记录准确的共享行为信息,而仅记录了共享者的数量,因而在目录项的替换时,需要对共享者进行整个区域范围的无效操作,造成了一定的性能损失。不过目录项替换发生的频率较低,而且替换时可以选择替换代价较小的目录项进行替换,如块目录项。如果必须进行替换,可以通过在私有缓存中支持区域无效操作,仅向每个私有缓存发送一条区域无效命令即可完成整个缓存区域的无效操作,从而加速替换过程。

4 实验数据

4.1 实验平台

本文采用 MIT 大学发布的 Graphite 分布式多核模拟器[13],Graphite 可以直接在主机上运行被模拟的程序,基于动态插桩的方式获取执行过程中的信息,模拟目标结构的运行机制,从而获取性能等参数。该模拟器的特性是系统开销小,可并行模拟多核结构,速度较快,但是由于它不是时钟精确类型的模拟器,对于模拟要求具有精确时钟信息的结构准确性较差。不过本文主要评估多核处理器的访存行为,时钟上的偏差对访存行为的定性结果影响不大,因此该模拟器能够达到本文的实验要求。

本文主要研究一致性目录结构的设计空间以使得缓存一致性协议可以更高效地运行,同时能够节省目录存储空间。而处理器微结构以及缓存层次的组织结构则不在考察的范围内,因此仅采用比较常用的设计参数。具体的参数配置如表 1 所示。

表 **1** 目标系统的参数配置

结构名称	参数配置
处理器核	按序发射,1GHz,16 核
L1 缓存	私有,32KB 数据,32KB 指令,4 路组相联,2 时钟周期,64 字节缓存块
L2 缓存	共享,256KB,16 路组相联,10 时钟周期,64 字节缓存块
互连网络	4×4 Mesh 结构,128bit 位宽,每链路传输延时 2 时钟周期
内存	200 时钟周期

4.2　目录性能评估

在某个缓存区域的生命期中,缓存区域的共享者数目和被共享的缓存块数目的不同会影响其所需创建的目录项数。对于 DGD 来说,非区域拥有者访问的缓存块的数量决定了需要额外创建的块目录项的数量。本文用每缓存区域平均需要创建的目录项数(Average Directory Entry Created, ADEC)来评估目录的空间利用率。

如图 7 所示,RSDGD 和 DGD 相比,缓存区域的 ADEC 在各测试程序下有不同程度的降低。如 ocean_ continous, water-spatio, barnes 等测试程序减少了 40% ~45% 的 ADEC,其他大部分测试程序均降低了 20% ~30%,而 water-nsquared, raytrace 的优化效果较不明显,ADEC 降低了约 12% 左右。主要是由于这些应用的共享者数量大于等于 2 个的缓存区域数目较少,DGD 已经能够较好地处理一个共享者的情形,RSDGD 带来的好处有限。总的来说,测试程序集平均可以减少 25% 的目录空间需求。

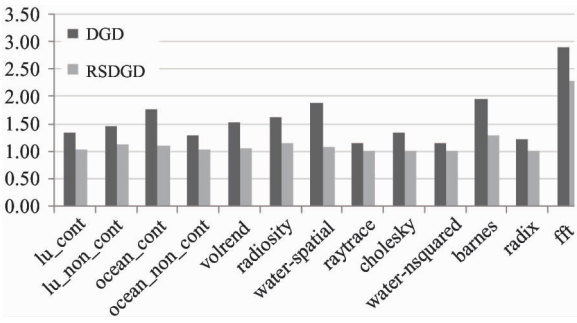

图 7　缓存区域平均创建的目录项数(**ADEC**)

4.3　错误率以及性能开销评估

图 8 给出了平均每 10000 条一致性消息中的冗余共享消息数量,RSDGD 在创建块目录项时将区域共享者默认作为缓存块的共享者的策略对互连网络增加了仅 0.6% 的网络消息,对性能的损失微乎其微。一方面是因为创建的块目录项大大减少了;另一方面是因为一部分区域共享者实际也共享了该缓存块。

5　结　论

本文评估了基于片上多核处理器结构的双粒度目录结构的共享行为,并在双粒度目录框架下提出了区域共享的双粒度目录结构,使其在保留原有的双粒度目录方案的优点之外,更好地处理多个共享者缓存区域的访存行为。该结构相比原有的双粒度目录结构,所需的目录项数更少,而且由于降低了所需创建的目录项,避免目录项替换等性能开销,提高了访存性能。未来的工作包括:进一步优化目录结构,利用访存行为的时空特性,使得区域目录项能够支持更多的区域访存行为;优化区域目录项中共享者的插入退出机制;利用目录中区域目录项的信息,优化共享缓存的压缩性能,进行对 IO 访问的优化。

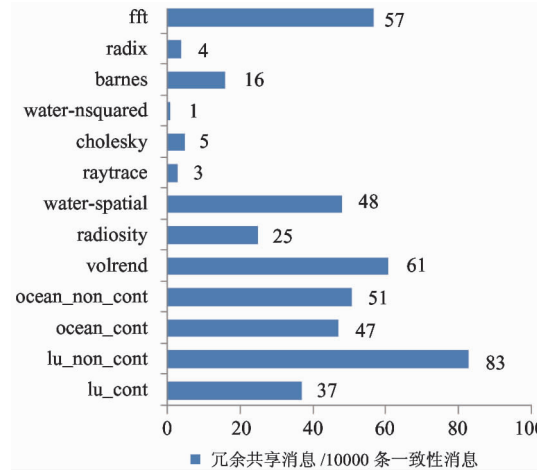

图 8　冗余共享者的开销评估

参考文献

[1] Agarwal A, Simoni R, Hennessy J, et al. An evaluation of directory schemes for cache coherence. *ACM SIGARCH Computer Architecture News*, 1988, 16: 280-298

[2] Chaiken D, Kubiatowicz J, Agarwal A. LimitLESS directories: a scalable cache coherence scheme. In: Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, Santa Clara, USA, 1991. 224-234

[3] Choi J H, Park K H. Segment directory enhancing the limited directory cache coherence schemes. In: Proceedings of the 13th International Symposium on Parallel Processing and the 10th Symposium on Parallel and Distributed Processing, San Juan, Puerto Rico, 1999. 258-267

[4] Zebchuk J, Qureshi M K, Srinivasan V, et al. A tagless coherence directory. In: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, New York, USA, 2009. 423-434

[5] Ferdman M, Lotfi-Kamran P, Balet K, et al. Cuckoo directory: A scalable directory for many-core systems. In: Proceedings of the 17th International Conference on High-Performance Computer Architecture, San Antonio, USA, 2011. 169-180

[6] Sanchez D, Kozyrakis C. SCD: A scalable coherence directory with flexible sharer set encoding. In: Proceedings of the 18th International Conference on High-Performance Computer Architecture, New Orleans, USA, 2012. 1-12

[7] Gupta A, Weber W D, Mowry T C. Reducing memory and traffic requirements for scalable directory-based cache coherence schemes. In: Proceedings of the 1990 International Conference on Parallel Processing, Urbana-Champaign, USA, 1990. 312-321

[8] Kelm J H, Johnson M R, Lumettta S S, et al. Waypoint: scaling coherence to thousand-core architectures. In: Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques, Vienna, Austria, 2010. 99-110

[9] Cuesta B, Ros A, Gomez M E. Increasing the effectiveness of directory caches by avoiding the tracking of noncoherent memory blocks. *IEEE Transactions on Computers*, 2013, 62(3): 482-495

[10] Pugsley S H, Spjut J B, Nellans D W, et al. Swel: hardware cache coherence protocols to map shared data onto shared caches. In: Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques, Vienna, Austria, 2010. 465-476

[11] Alisafaee M. Spatiotemporal coherence tracking. In: Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture, Vancouver, Canada, 2012. 341-350

[12] Zebchuk J, Falsafi B, Moshovos A. Multi-grain coherence directory. In: Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture, Davis, USA, 2013. 359-370

[13] Miller J E, Kasture H, Kurian G, et al. Graphite: A distributed parallel simulator for multicores. In: Proceedings of the IEEE 16th International Symposium on High Performance Computer Architecture, Bangarlore, India, 2010. 1-12

A region shared dual-grain directory for chip multi-core processors

Zeng Lu[*,**,***], Chen Xinke[*,**,***], Wang Huandong[****]

([*] State Key Laboratory of Computer Architecture (Institute of Computing Technology, Chinese Academy of Sciences), Beijing 100190)

([**] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([***] University of Chinese Academy of Sciences, Beijing 100049)

([****] Loongson Technology Corporation Limited, Beijing 100190)

Abstract

The memory access behavior of chip multi-core processors under the dual-grain directory (DGD) structure and DGD's directory overhead on sharing behavior were studied to decrease the area overhead of the DGD structure. Aiming at the problem that DGD needs to create additional block directory entries for shared cache regions, an innovatory region shared DGD, called RSDGD, was proposed. The proposed directory can use one region shared directory entry to track coherence information of a memory region for at most 3 sharers, thus the additional block directory entries are efficiently reduced, and the directory area is saved. The experimental results show that the region shared DGD can save the directory area by 25% in average compared with the former DGD solution, while the caused performance lost is just under 0.6%. This solution is proved to be effective in reducing chip area and increasing directory structure's elasticity.

Key words: dual-grain directory (DGD), chip multi-core processor, cache coherence, region shared, directory coherence protocol, memory access optimization